Multiple target tracking for intelligent headlights control

Jose C. Rubio, Joan Serrat, Antonio M. López and Daniel Ponsa

***Abstract*— Intelligent vehicle lighting systems aim at automatically regulate the headlights' beam angle so as to illuminate as much of the road ahead as possible, while avoiding dazzling other drivers. A key component of such a system is a computer vision software able to distinguish blobs due to vehicles' head and rear-lights from those originating from road lamps and reflective elements like poles and traffic signs. In a previous work, we have devised a set of specialized supervised classifiers to make such decisions based on blob features related to its intensity and shape. Despite the overall good performance, there remain challenging cases not yet solved which hamper the adoption of such a system; notably, faint and tiny blobs corresponding to quite distant vehicles which disapear and reappear now and then. One reason for the errors in the classification is that it was carried out independently of other frames. Hence, we address the problem by tracking blobs in order to 1) obtain more feature measurements per blob along its track, 2) compute motion features, which we deem relevant for the classification and 3) enforce its temporal consistency. This paper focuses on the problem of constructing blob tracks, which is actually one of multiple target tracking, but under special conditions: we have to deal with frequent occlusions as well as blob splitings and mergings. We approach it in a novel way, by formulating the problem as a maximum a posteriori inference on a Markov random field. We present qualitative (in video form) and quantitative results which show that our new tracking method achieves good tracking results with regard to the original objective.**

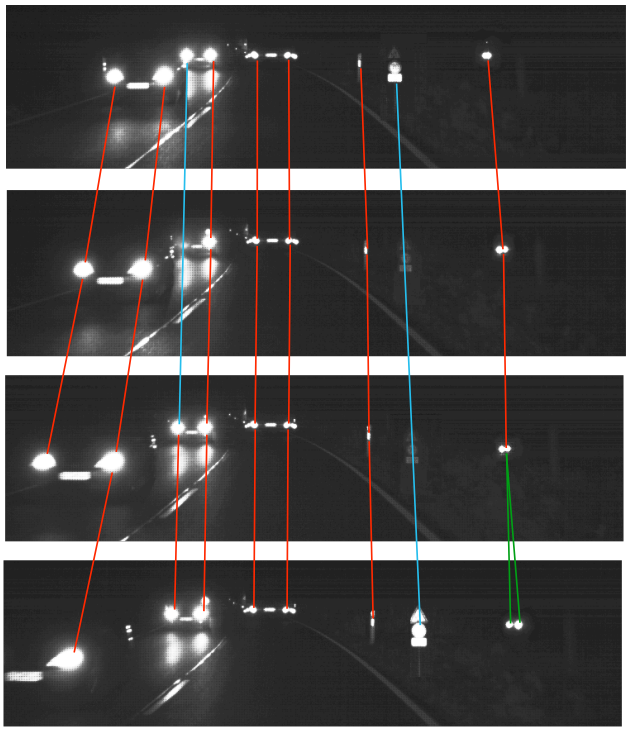


Fig. 1. Example of blob tracking along four frames. In red, one–to–one associations, in blue two occlusions and in green a splitting.

I. INTRODUCTION

Accident statistics demonstrate that driving at night is considerably more dangerous than its daytime counterpart [1]. This can be attributed, among other causes, to the lower performance of the human visual system under poor ambient lighting conditions: color and depth perception, and therefore object saliency, are reduced. Some studies like [2] show that drivers turn on high beams much less frequently than they can: only about one fourth of the time during which traffic conditions would justify their use. Among the reasons for this behavior, we highlight two: the need for a manual (and eventually, frequent) operation and the fear of dazzling drivers of leading, oncoming or overtaking vehicles. Recently, the combination of specialized on-board cameras, fast processors and machine learning techniques has enabled some automotive machine vision suppliers and companies to develop prototypes of 'intelligent headlights' controllers (IHC) for high-end car series, with acceptable results. However, as we will discuss, this problem is far from being completely solved.

The core of an IHC is pattern classification software able to discern bright spots (or image blobs, in computer vision terminology) originating from vehicles' head or rearlights from those due to road lamps, traffic-lights and reflective infrastructure elements like poles, lane markings and traffic signs. The main difficulties are due to the requirements of a low classification error rate, real-time processing and, perhaps more importantly, the need to detect all vehicles whithin the image field of view as soon as possible. This poses a problem in the case of very distant vehicles. Vehicles are considered distant at 600 meters for oncoming vehicles and 400 meters for leading vehicles, because of the different glaring effect of the host vehicle high-beams on their drivers. At these distances, head/rear-lights are imaged as tiny blobs, fewer than ten pixels in size, so that appearance features such as intensity, color and shape do not provide sufficient information to perform a reliable classification of individual blobs *when frames are examined independently*.

The literature on nighttime, on-board, vision-based vehicle

This work was supported by the Spanish Ministry of Education and Science under Project TRA2007–62526/AUT and the Research Program Consolider Ingenio 2010: MIPRCV (CSD2007–00018)

Authors are from the Computer Vision Center and the Dept. of Computer Science, Universitat Autònoma de Barcelona, 08193 Cerdanyola, Spain. jcrubio@cvc.uab.es

detection is rather scarce. All the works reviewed first segment the image by some variant of adaptive thresholding, then perform the classification based on features related to color, shape, size and image location. The simplest classification methods use a set of heuristic rules with fixed thresholds [3]–[6]. Other works employ machine learning techniques like decision trees [7], Bayes factors [8], Support Vector Machines [9]–[12] and Real-AdaBoost [13], [14], which can be trained and thus possess much greater adaptability. Some of these works recognize that the classifier outcome is not sufficiently reliable and that decisions for one blob are not stable along time. To remedy this, they either track blobs or pair them as belonging to the same vehicle.

Pairing, or more generally, clustering blobs, helps to better classify blobs since only those that form a consistent pair, according to constraints like similar vertical position, size, shape, color etc., can originate from a vehicle [6]. However, this is not a convenient strategy if vehicle detection has to be used for IHC, because the two head or rearlights appear separated in the images once they are close to the host vehicle. Hence, distant vehicles would never be reconized as such. Pairing suits vehicle detection for other driving assistance applications, like estimation of time to collision [3]–[5], [8], [12] or automatic cruise control. In these two cases, the separation of spotlights is necessary to estimate the distance to other vehicles.

Therefore, tracking seems the only way left for an IHC to avoid the errors induced by the frame-by-frame independent classification. Specifically, the potential benefits of tracking blobs for IHC are its ability to

- increase the number of feature/attribute measurements of each tracked blob;
- provide the classifier with additional motion features [6], [9], [10], otherwise not available;
- allow the selection of 'interesting' blobs which are passed to the classifier as those that can be followed during a certain minimum number of frames [7], [8], [11], [12];
- associate a confidence to the class label of a blob — high if it is consistent with labels of past frames, low if not— and make the final classification decision at the moment its confidence exceeds a certain threshold [13], [14].

Few works on IHC perform tracking, probably due to its difficulty in this context. In [7]–[9] a simple nearest neighbor search is performed, based on image location and appearance features. This is also done in [6], [10]–[12] with individual or clusters of blobs, though they first predict the position of blobs by means of a Kalman filter. In both cases, tracking refers to associating blobs from one frame to the next. In [14] proper tracking is replaced by a so-called 'temporal coherence analysis' whereby a confidence map is maintained, quantifying the belief in finding a vehicle blob at each pixel. This confidence is estimated on the basis of the blob labels at the frames immediately preceding the current frame. Despite fostering the temporal coherence of the classification, this method does not produce blob tracks.

In order to take full advantage of the potential benefits discussed previously, the tracking algorithm must deal sucessfully with blob occlusions, splittings and mergings. Occlusion handling means that blobs which temporally disappear must not originate new tracks but be associated with their former track. Splittings occur when a blob corresponding to the two headlamps of an oncoming vehicle becomes two distinct blobs as the vehicle approaches. Splittings may also occur with static reflective surfaces like poles. Merging is the opposite case: as a leading vehicle (or a compound traffic sign) gets farther away, two blobs merge into a single one. These are frequent events in nighttime video sequences, and are caused by distant vehicles, light sources or reflections not directed towards the camera and distant, small or poorly reflective surfaces. In spite of their importance, none of the reviewed works which perform tracking deals with them.

This paper introduces two main contributions. First, we focus on the problem of building tracks of close, mid-distance and far away light sources/reflectors taking into account occlusions, merges and splits. In particular, we solve the problem of building continuous tracks in the presence of occlusions up to a certain duration. Second, we propose a new probabilistic tracking method whereby the problem is posed as a maximum a posteriori estimation in a Markov random field. Associating two blobs from different frames within a certain time window is represented by a binary variable whose most probable state, either associated or not, must be estimated. Once a solution is found for every association for a time window, we propagate the result to the next frame by sliding the window. We provide extensive quantitative evaluations, based on annotations of tracks on five video sequences. We also include qualitative results in video form as additional material of this manuscript.

In section II we introduce the probabilistic model. In III, the estimation propagation within windows is explained, and in IV we present the inference algorithm used. Finally, in section V we show the algorithm results, and in VI we draw conclusions.

II. PROBABILISTIC MULTIPLE FRAME ASSIGNMENT

Let w be the number of contiguous frames in a certain temporal window of the video sequence in which we want to track point features. We denote by $I_1, I_2, ..., I_w$ the different frames within it. Each frame contains a set of zero or more point features, indexed by $p, q, ...$. An association a is an ordered pair of features from different frames, $a = (p, q)$, meaning that features p and q are observations (sensor measurements) from the same target, but at different frames. Let A be the set of all such associations,

$$A = \{a = (p, q) | p \in I_i, q \in I_j, 1 \leq i < j \leq w\}, \quad (1)$$

where $a, b, ...$ index the elements of A, so that we can denote all pairs of association without repeated combinations as $(a, b), a < b$. Let $\mathbf{X} = (...X_a...)$ be the vector of binary variables, one per association, where $X_a = 1$ if the

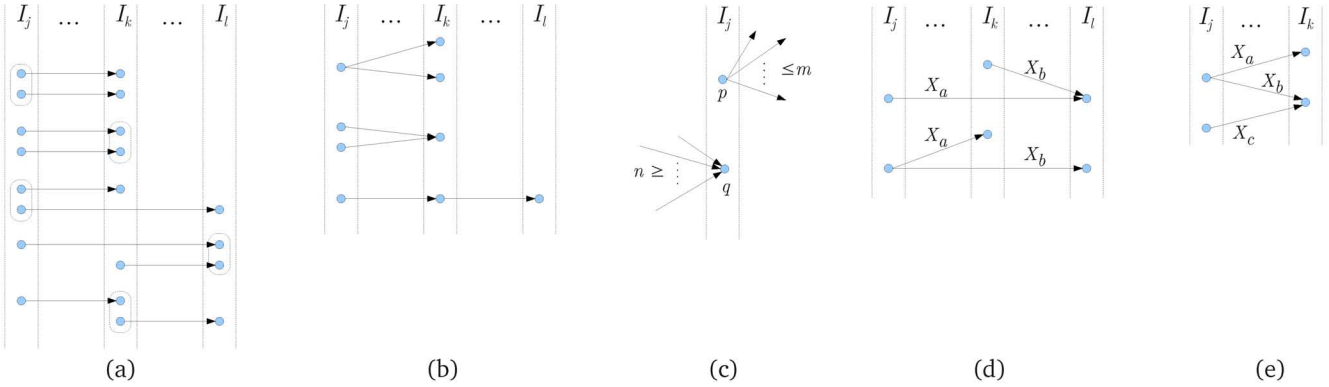


Fig. 2. Types of associations involved in the (a,b) likelihood term and (c,d,e) the prior terms. Vertical strips represent frames, circles features and arrows associations between two features. Dashed ellipses mean neighbour features. Note that (a) and (b) are *all* possible types of association pairs: in (a) from top to down, pairs leaving or arriving to neighboring features in the same frame, pairs leaving neighboring features but arriving to features in different frames, and pairs arriving to neighboring features in the same frame, but leaving from different frames. Also, the last row illustrates a pair where one association arrives to a feature, which is neighbor of the origin feature of the pairing association. In the last row of (b) pairs sharing a feature either at the origin, destination or intermediate frame. The constraint $X_a + X_b \leq 1$ on the association pairs in (d) precludes other pair possibilities.

corresponding association a exists, and zero otherwise. In the same way, the vector of all observations is denoted by $\mathbf{Y} = (...Y_a...)$, where each association $a = (p, q)$ is represented by $Y_a = [p_x, p_y, q_x, q_y, p_{area}, q_{area}]$. Thus, each observation is a vector of properties: the spatial coordinates and areas of its origin and destination feature points. Although, other properties may be also considered, like shape or intensity measures.

Our goal is to find the most likely configuration of the set $\mathbf{X}$ of association states, given the set of all observations $\mathbf{Y}$. This is, to find the maximum a posteriori estimation,

$$\mathbf{X}^* = \arg\max_{\mathbf{X}} \mathbf{p}(\mathbf{X}|\mathbf{Y}). \tag{2}$$

In a Bayesian framework, the posterior probability of the the hidden variables $\mathbf{X}$, given the observations, is proportional to the product of the likelihood and prior terms

$$p(\mathbf{X}|\mathbf{Y}) \propto p(\mathbf{Y}|\mathbf{X})p(\mathbf{X}). \tag{3}$$

The likelihood term $p(\mathbf{Y}|\mathbf{X})$ encodes the application specific observation model. The prior $p(\mathbf{X})$ encodes de application restrictions. The next two sections detail how do we define and compute these two terms.

A. Likelihood

We assume the observation model $p(\mathbf{Y}|\mathbf{X})$, factorizes as

$$p(\mathbf{Y}|\mathbf{X}) = \left[\prod_{a\in A} p_A(Y_a|X_a)\right] \cdot \left[\prod_{(a,b)\in N} p_N(Y_a, Y_b|X_a, X_b)\right] \cdot \left[\prod_{(a,b,c,...)\in O} p_O(Y_a, Y_b, Y_c, ...|X_a, X_b, X_c, ...)\right]. \tag{4}$$

The first term models the likelihood of an association being active or inactive, depending on the similarity (area, location) of the two features (p, q) involved in each association $a \in A$. The second term is the likelihood of two associations existing simultaneously. This exploits the spatial relationships of the observations of two associations, imposing a local invariance to rotation and translation, and it is defined over the set N of all association pairs, as will be explained below. The third component represents the probability of a feature being occluded, during one or more frames, defined over the set O, which contains groups of associations involved in the occlusion of a feature. This term is needed because it plays the key role of avoiding the trivial solution $X_a = 0\ \forall a \in A$, as we will discuss later. Next, we describe the probability models for each term of the factorization of Eq. (4).

Appearance. The displacement of a feature between two frames and its position are not independent. As the feature approaches the camera, it moves towards the image left and right borders, and its apparent velocity increases. In contrast, it remains motionless when distant, positioned in the center of the image. Moreover, the feature position has a direct relationship with its area: the closer a feature is to the camera, the faster its area changes from one frame to another. Accordingly, an association $a = (p, q)$ is more probable if the areas of p and q are similar, and their positions change as described. We define the probability of associating feature p with feature q as

$$p_A(Y_a|X_a = 1) = \hat{f}_1(\mathbf{v}_{pq}, p_x)\hat{g}_1(|p_{area} - q_{area}|, p_x)). \tag{5}$$

$$p_A(Y_a|X_a = 0) = \hat{f}_0(\mathbf{v}_{pq}, p_x)\hat{g}_0(|p_{area} - q_{area}|, p_x)). \tag{6}$$

$\hat{f}_1$ is a density function modeling the dependency of the location and the displacement vector $\mathbf{v}_{pq}$ of the association a. The density $\hat{g}_1$ models the relationship between the original position of the feature and the frame-to-frame change in the areas of the features. Analogously, $\hat{f}_0$ and $\hat{g}_o$ define the same correlations for the case of a not existing.

All these probability densities are learned using a Kernel Density Estimator (KDE) [15]. The expression of $\hat{f}$, corresponds to the well known bivariate kernel density estimator

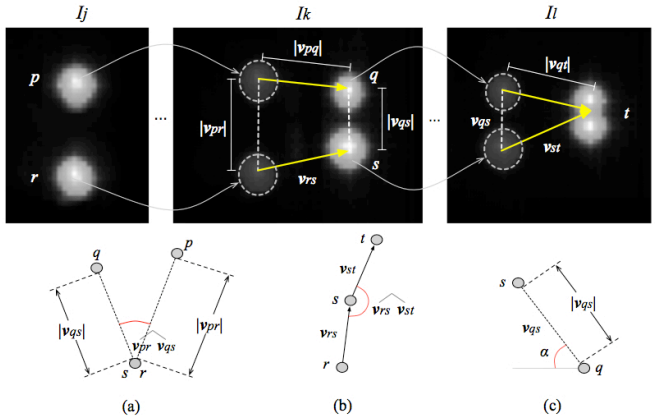


Fig. 3. Example of the different measures used when calculating the probability components. a) represents the angle and distance used in p_G. b) represents the angle when enforcing linear trajectory in the p_M. and c) represents the module vector when calculating the merging & splitting compatibility p_{SM}, and the angle against an horizontal reference vector.

$$\hat{f}(x,y) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_x h_y} K\left(\frac{x-x_i}{h_x}, \frac{y-y_i}{h_y}\right), \quad (7)$$

where K is a Gaussian kernel, h_x, h_y are the bandwidth components, which are data-driven and automatically selected, and n is the number of training data points.

The term p_N is defined over the set N of pairs of associations,

$$N = \{(a,b) \in A \times A | a < b\}, \quad (8)$$

corresponding to Fig. 2a and the last row of 2b. The term p_N is responsible for modeling relationships between pairs of associations. We distinguish three components into which p_N factorizes, which we call *Geometry*, *Motion* and *Split & Merge*.

$$p_N(Y_a, Y_b | X_a, X_b) = p_G(Y_a, Y_b | X_a, X_b) \cdot p_M(Y_a, Y_b | X_a, X_b) p_{SM}(Y_a, Y_b | X_a, X_b). \quad (9)$$

Each of these terms defines the likelihood of different pairs of associations. For instance, two associations in consecutive frames, as illustrated in the third row of Fig.2b are involved in the *motion* likelihood. On the other hand, two associations like the ones in the first row of Fig.2a relate to the *geometric* likelihood.

Geometry. Targets do not move independently of each other. Two close targets are likely to move in a similar direction and with a similar speed. This can be seen as a local isometric mapping of the points from one frame to another. This means enforcing invariance on the distance and angles defined by pairs of points. See Figure 3a and 3b. Two associations are probable if the distance between their origin features is very similar to the distance between their destination features, and if the angle between the origin features is similar to the angle between the destination features.

The likelihood p_G is defined over the set of pairs of features:

$$N_G = \{(a,b) \in N | a = (p,q), b = (r,s), q \in N_p \vee p \in N_q \vee r \in N_s \vee s \in N_r\}, \quad (10)$$

where N_p defines the set of neighbors of p, according to some neighborhood definition. In our case it is the set of k-nearest features of p, whose distance to it is under a certain threshold.

Let $a = (p,q)$ and $b = (r,s)$ be two associations of N_G, X_a, X_b their states, and Y_a, Y_b their respective observations. Let $\mathbf{v}_{pr}$ be the vector between their origin features, and $\mathbf{v}_{qs}$ the vector between the destination features, as shown in Fig. 3 (top). Since we assume these measures to be independent, we can construct the probability as a convex combination of Gaussian densities, as follows:

$$p_G(Y_a, Y_b | X_a = 1, X_b = 1) = \lambda_G \mathcal{N}(\widehat{\mathbf{v}_{pr}\mathbf{v}_{qs}}) + (1-\lambda_G)\mathcal{N}(|\mathbf{v}_{pr}| - |\mathbf{v}_{qs}|), \quad (11)$$

where $\mathcal{N}(x)$ represents a normal distribution. For the sake of readability we represent the gaussians as $\mathcal{N}(x)$, instead of $\mathcal{N}(x; \mu, \sigma^2)$, for some μ, σ. Thus, we learn their parameters, μ, σ^2 from training data, using the standard method of maximum likelihood. The parameter $\lambda \in [0,1]$ weights the contribution of each term to the mixture of densities. The first normal distribution measures the similarity between the orientation of $\mathbf{v}_{pr}$ and $\mathbf{v}_{qs}$, and the second enforces the similarity between $|\mathbf{v}_{pr}|$ and $|\mathbf{v}_{qs}|$.

Motion. Close targets tend to follow a linear trajectory when their position is close enough to the camera, while far away targets, imaged around the image center, are static or oscillate up and down due to the movement of the vehicle where the camera is installed. The set of pairs related to the motion component correspond to the third pattern of Fig. 2b, and is defined as

$$N_M = \{(a,b) \in N | a = (p,q), b = (r,s), q = r\}. \quad (12)$$

Two associations in the motion set N_M are probable if, while close to the camera, the displacement of their features follow a similar direction. Given two associations $a = (p,q)$ and $b = (q,r)$ from N_M, the likelihood of these associations existing simultaneously is defined as

$$p_M(Y_a, Y_b | X_a = 1, X_b = 1) = \hat{h}(\widehat{\mathbf{v}_{pq}\mathbf{v}_{rs}}, q_x), \quad (13)$$

where the density $\hat{h}$, depends on the angle $\widehat{\mathbf{v}_{pq}\mathbf{v}_{rs}}$ (3c), and the horizontal position of the central feature q. This encourages similarity of the vector directions of pairs of associations close to the image's left and right borders. Again, the correlation defined by $\hat{h}$ cannot be modeled by a simple Gaussian. A non-parametric Kernel Density Estimator is used to learn the density shape from training data.

Split and Merge. This term models the probability of two features merging, or one feature splitting in two. Given two

associations $a = (p, q)$ and $b = (r, s)$, a splitting occurs when $p = r$, and a merging when $q = s$. The set of pairs belonging to the split and merge component follow to the first and second patterns of Fig. 2b, and is defined as

$$N_{SM} = \{(a, b) \in N | a = (p, q), \quad b = (r, s), (p = r) \vee (q = s)\}. \tag{14}$$

The merging or splitting features are likely to have similar areas and to have very close positions. In addition, we restrict the mergings and splittings to only horizontal or vertically. This restriction reflects the nature of the merging and splittings that are originated by road pole reflections (vertical) or car headlights (horizontal). The probability of a merging or splitting is defined as

$$\begin{aligned} &p_{SM}(Y_a, Y_b | X_a = 1, X_b = 1) \\ &= [\lambda_{SM}\mathcal{N}(|\mathbf{v}_{pr}|) + (1 - \lambda_{SM})\mathcal{N}(|p_{area} - r_{area}|)] \\ &\cdot \mathcal{N}(|\pi/4 - \alpha|). \end{aligned} \tag{15}$$

The first two distributions form a Gaussian mixture, whose components are weighted by the parameter λ_{SM}. The first demands merging or splitting when the distance between the targets $|\mathbf{v}_{pr}|$ is small, while the other favors area similarity. Figure 3c shows an example. The angle α is the angle between vector $\mathbf{v}_{qs}$ and a reference horizontal vector. Hence, the last distribution enforces horizontal and vertical alignment of the targets which are merging or splitting.

Occlusions. We say a feature p has been occluded along d frames when p, being visible in frame I_i, disappears during d consecutive frames to either appear again in frame I_{i+d}, or disappear definitively from the window. Let $A(p, d)$ be the set of all associations with origin in feature $p \in I_i$, and with destination features located in frames $I_{i+1}, I_{i+2}, ..., I_{i+d}$. This is defined as

$$A(p, d) = \{a = (p, q) \in A | p \in I_i, q \in I_{i+k}, k = 1...d\}. \tag{16}$$

The feature p is considered occluded with duration d when every $a \in A(p, d)$ is zero. Therefore, the conditional probability which models an occlusion depends on all the associations departing from p and having duration d. The set of associations which define an occlusion, for all possible durations within a window of w frames, is

$$O = \{A(p, d) | d = 1...w - 1, i = 1...w - d, p \in I_i\}. \tag{17}$$

The probability distribution p_O is built around two assumptions. First, the features close to the borders of the image are more likely to disappear. This is due to the movement of the blobs, which typically appear in the image center, and then move towards the image borders. Although this does not always happens, because of leading and overtaking vehicles, which move faster than ours. Second, tiny features are also likely to disappear, as a consequence of the segmentation process. Finally, we encourage features to be associated with other features belonging to the closest frame possible. Put all together, for each $\{a, b, ...\} = A(p, d) \in O$,

$$\begin{aligned} &p_O(Y_a, Y_b, ... | X_a, X_b, ...) \\ &= \begin{cases} \hat{i}(p_x, p_{area})(1 - e^{-d}) & \text{if } X_c = 0 \ \forall c \in A(p, d) \\ 1 & \text{otherwise} \end{cases} \end{aligned} \tag{18}$$

The density distribution $\hat{i}$ models the probability of all associations coming from a feature p with duration d being inactive. The term $(1 - e^{-d})$ favors associations between features in nearby frames.

Note that if at least one association X_a does exist, $p_O = 1$. In the same way, every conditional distribution of the terms p_G, p_M and p_{SM} explained thus far, with the exception of the appearance term, depends on a specific realization of the random variables $(X_a = 1, X_b = 1)$. It is important to notice that the probability values for the rest of configurations, $(X_a = 0, X_b = 0), (X_a = 0, X_b = 1)$, and $(X_a = 1, X_b = 0)$ are set to one, which means that the observation model does not 'penalize' these realizations. Hence, why the trivial solution $\mathbf{X} = (0, ..., 0)$ is not the most probable?. The appearance and the occlusion term, are in charge of avoiding a trivial solution in which every variable state is zero. For instance, if a feature p is not likely to be occluded in the next d frames, it will assign a low probability to the configuration $X_a = 0, X_b = 0, ...$, where $(a, b, ...) \in A(p, d)$.

B. Modeling the Prior

We include a constraint on the maximum number of features to which one feature can be associated. This may be used in tracking applications for which we know the bounds on the number of features involved in splits and merges. This set of constrains is shown in Fig. 2c. Given two frames I_i, I_j, from a window of length w, we define what we call the multi-assignment m-to-n constraint as

$$\sum_{a \in A(p)} X_a \leq m, \forall p \in I_i, i = 1 \dots w - 1 \tag{19}$$

$$\sum_{b \in B(q)} X_b \leq n, \forall q \in I_j, j = 2 \dots w, \tag{20}$$

where $A(p)$ is the set of associations leaving feature $p \in I_i$ and $B(q)$ the set of those arriving at $q \in I_j$. In our case, $m = n = 2$, meaning that we restrict the number of targets merging or splitting to a maximum of two. For instance, when headlights or rear-lights merge or split.

Split and merge handling gives rise to two additional sets of constraints. The first, corresponding to Fig. 2d, comes from the condition that splits and merges occur in precisely two frames I_i, I_j. It takes the form

$$X_a + X_b \leq 1, \tag{21}$$

for all pairs $(a, b), a < b$ such that if $a = (p, q), b = (r, s)$ and $i < j < k$ then either $p = r \in I_i$, $q \in I_j$, $s \in I_k$ or $q = p \in I_i, r \in I_j, s \in I_k$.

The second set of constraints expresses the assumption that a merge cannot mix with a split and vice versa, as Figure

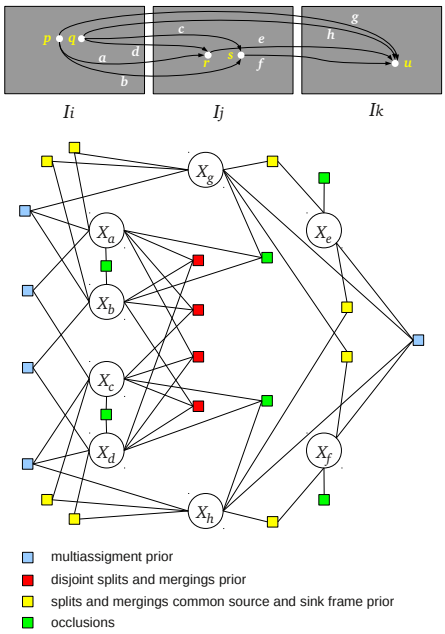


Fig. 4. Example of factor graph for a window of three frames containing five blobs. Variables $X_a \dots X_h$ represent the eight possible associations $a = (p, r), b = (p, s), \dots h = (r, u)$. Only represented factors to model occlusions and the three components of the prior: multi-assignment limits, splits and mergings restricted to two frames, and split and mergings on disjoint sets of features.

2e illustrates. For features within the same frame, the set of features involved in a split are disjoint from those involved in a merge. This takes the form

$$X_a + X_b + X_c \leq 2, \tag{22}$$

where $a = (p, q), b = (p, s), c = (t, s)$ and $p, t \in I_i$, q, $s \in I_j$, for all $1 \leq I_i < I_j \leq w$.

Note that all the constraints of Eqs. (19) - (22) have the form of an upper bound on a linear combination of a few association variables. Thus, if r is the number of constraints, all of them can be compactly expressed as $C\mathbf{X}^T \leq \mathbf{b}$, where $C = [\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_r]^T$ is a very sparse binary matrix whose rows select the variables of each constraint, and $\mathbf{b}$ is a column vector with bounds m, n, 1 and 2. Then, the prior reduces to

$$P(\mathbf{X} = \mathbf{x}) = \begin{cases} 1 & \text{if } C\mathbf{x} \leq \mathbf{b} \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

III. FROM WINDOW ASSIGNMENT TO SEQUENCE TRACKS

So far, we have explained how to track features within a window of w frames. Next, we will see how to extend the algorithm in order to track long sequences

Every association which originates in the first frame of a window t, and is set as 'existing' by the inference algorithm, is added to the final track results. Doing this, the algorithm is able to recover every occlusion whose duration is within the window size. With the aim of recovering the maximum amount of occlusions possible, the sliding step of the window is set to one frame. Moreover, all these associations are introduced as new observations in the next sliding window $t + 1$, incorporating inference information from the past window. The process is repeated up to the last w frames of the sequence, when all active associations from the last window are added to the final track result.

Inter-window (IW) information can be easily added to the probabilistic model, as mentioned earlier, by including the active associations obtained in window t as observations in the factor graph of window $t+1$. This gives rise to a new term which is included in the likelihood factorization, analogous to the motion component p_M, explained in Eq. (13). This term enforces linear trajectories in the movement of the features, but in this case within consecutive windows. It is defined as:

$$p_{IW}(Y_a, Y_b | X_b) = p_M(Y_a, Y_b | X_a = 1, X_b). \tag{24}$$

IV. APPROXIMATE INFERENCE WITH BELIEF PROPAGATION

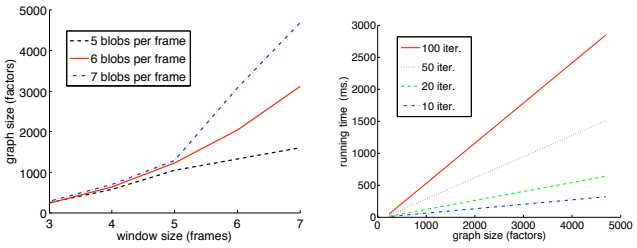
Searching for an optimal vector $\mathbf{X}$ which maximizes the expression in Eq. (2) is, in general, NP-hard. In this paper we use the max-product algorithm to calculate an approximation of the MAP configuration of the vector of random variables, on a Markov Random Field formed by the variables X_a, the observations Y_a, for all $a \in A$, and the factor functions defined by Equations (5) to (23).

The max-product Belief Propagation [16] is a message-passing algorithm on factor graphs, known to converge to a fixed point when the graph is a tree. Although the resulting graph in this work does not have this structure, as can be seen in Figure 4, several researchers have recently reported excellent results by running max-product algorithms on graphs with loops [17]. The running time for Belief Propagation is $\mathcal{O}(MN^k)$, where M is the number of random variables, N is the possible labels for each variable, and k is the size of the maximum clique (number of variables involved in a factor). In order to preserve the tractability of the algorithm, most of the authors keep the size of the cliques pair-wise, at the expense of expressive power in the variable dependencies.

In our case, the cliques are the sets of variables involved in each of the terms explained in section II-A. Therefore, the bigger cliques correspond to the multiplicity constrains of Eq. (19) and (20), and their maximum size can be very high, depending on the amount of features existing in each frame of the window. To deal with the high dimensionality of these terms, based on the work of [18] , we take advantage of the high degree of sparseness of the constraint functions of Eq. (19) and (20), to transform the high order clique into several quadratic cliques, by adding extra variables. Analogously, we apply the same procedure to other high order functions, such as the occlusions factor of Equation (18).

Figure 4 shows an example of a factor graph resulting from a window of three frames, displaying only the prior and occlusion factors. Each box represents a clique, which corresponds to the probability factors of Eqs. (18) and (19)-(22). An interesting question at this point is the size of the

graph, in terms of number of variables and factors. Supposing we have n features at each frame, the graph will have $w(w-1)n^2/2$ variables which is a huge number. However multiple tracking problems are inherently sparse: the vast majority of potential associations are quite unlikely, so that reasonable application-dependent heuristics (gating) can reduce them to a manageable size. We have used a simple distance threshold to discard unfeasible associations which join features that are very far away from each other.



(a) (b)

Fig. 5. a) shows the relationship between the window size and the size of the factor graph. b) represents the inference execution time as a function of the graph size, for different number of iterations

For the MAP inference we have used the C++ implementation of the max-sum algorithm from the libDAI library [19]. Processing a sliding temporal window —which is done for each frame— of 6 frames and 5 features per frame takes about 700 milliseconds on a 2.66 GHz Intel Core 2 Duo with 4GB of RAM. Figure 5 shows the relationship between the window size and the execution times. The running times exposed above are highly dependent on the number of iterations of the Belief Propagation algorithm, which is set to 100. Decreasing this number results in lower execution times, while gracefully degrading the quality of the results. This let us conclude that by reducing the number of iterations and keeping a moderate window size, the algorithm can reach the real time requirements of the application.

V. RESULTS

In this section we present quantitative results of experiments performed on real data. The video sequences were recorded by a camera with a CMOS image sensor from Aptima Imaging$^{(TM)}$ of 752×480 pixels of resolution. The lens, having a 40° angular field of view, makes the detection of distant taillights very challenging. For instance, a single taillight at 400 meters is imaged as a spot of 4 to 10 pixels.

Manually annotated ground truth data was used for training and evaluation of the tracking accuracy. The ground truth consists of blobs annotated with their corresponding track label, and contains 51 tracks, 8919 blobs, 54 occlusions, 47 merges and 60 splits. Merging or splitting targets are annotated in the ground-truth as such when they belong to the same object. We have trained the likelihood term with 600 frames extracted from 7 different sequences, and tested the method on 5 of these sequences, but on different frames. To evaluate the algorithm we have used simple metrics: the percentage of correctly labeled targets, recovered occlusions,

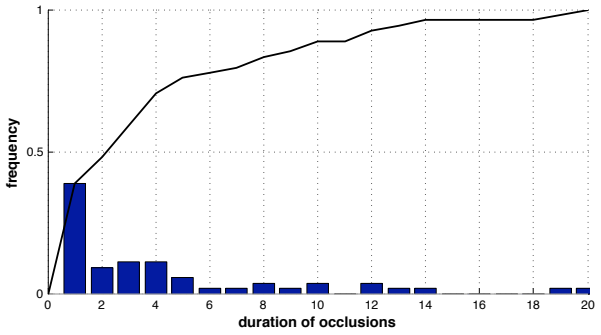


Fig. 6. Histogram and accumulated histogram of occlusion durations. A window of size w is able to recover all occlusions with duration up to $w-2$.

and merges and splits. To avoid counting the same error multiple times, a miss-detection of a merging or splitting is not considered as an incorrect labeling.

For all the experiments we have set parameters λ_G and λ_{SM} to 0.5. Recall that these parameters weight the contribution of each term of the mixture of densities for the Geometry and Split & Merge distributions respectively. A compromise was made between the length of the window, which dramatically influences the computation time, and the number of occlusions which can be recovered. We have counted the number of ground truth occlusions for different durations, as shown in Figure 6. The window length chosen for the experiments is 6, which is able to recover every occlusion of up to 4 frames of duration, or 70% of all occlusions.

TABLE I
TRACKING EVALUATION FOR SEQUENCES A TO E

	Sequences					
metrics in %	A	B	C	D	E	Mean
correct labeling	92	89	94	91	88	90.8
occlusions	60	57	65	71	63	63.2
mergings	36	63	51	47	37	46.8
splittings	68	58	67	59	61	62.6

Table I shows quantitative results of the tracking method. The percentage of correct labeling is over 90%, which illustrates the suitability of our method for tracking featureless tiny targets. The worst results were obtained for the detection of merging targets. This is due to the great difficulty when distinguishing a target which is merging or splitting from a target which is being occluded, or reappearing. For example, if two targets are very far from the camera and close to each other, and in the next frame there is only one target, it is very difficult, even for a human observer, to determine if the targets merged, or one of them has been occluded.

The occlusion recovery performs well. The fact that 63.2% of occlusions are recovered should be analyzed taking into account the number of occlusions which the method is able to treat with a window length of 6 frames, which is around 70%. Thus, the percentage of occlusions which are well-treated among the ones which are tractable is 90%.

We have constructed a web-page [20] where videos of 5

sequences with superimposed tracks can be viewed.

VI. CONCLUSIONS

We have shown that many-to-many feature matching can be applied to solve the problem of multiple target tracking, in the presence of target splits, merges and occlusions, obtaining high accuracy in real video sequences. We have developed a probabilistic model, in which the densities representing the application knowledge have been learned from training data. Tracking bright spots at night is known to be very challenging, especially for small features whose images have an area of less than 10 pixels. Our method is able to correctly track an average of 90% of such small blobs.

The main advantage of our method is its ability to encode complex relationships between the target characteristics, resulting in a flexible yet powerful model. We have introduced a novel explicit handling of occlusions, merges, and splits, creating continuous tracks of multiple targets. In IHC applications, this is necessary to extract multiple features from a blob along different frames, in order to improve the classification of difficult targets. However, the method can be easily extended to generic tracking applications.

Avenues for future research include: First, the merges and splits failures can be solved by increasing the amount of training data, and modeling a probability density which better suits the target behavior. Second, we would like to evaluate the improvements of the classification of [13], after incorporating our tracking algorithm.

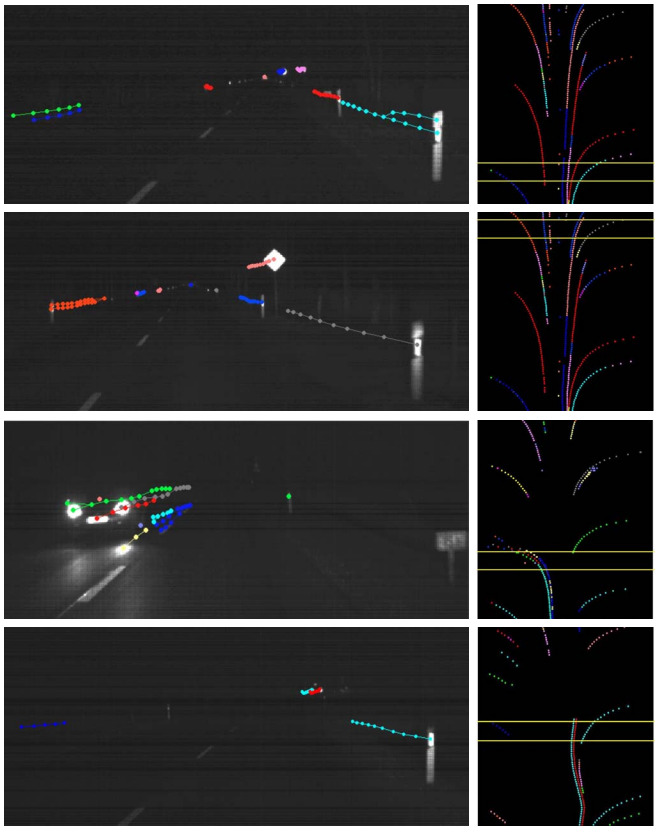


Fig. 7. Examples of resulting tracks obtained by our method. Each color indicates a different track identity. First column contains example frames, where, for the sake of visibility track lengths have been limited to the last 10 frames. Second column contains the complete tracks, represented in a plane, x position against time, increasing upwards A yellow horizontal stripe represents the time span in which the left snapshot is taken. First and second frames contain distant small targets and splittings of close road-poles. The third shows a close vehicle generating multiple tracks due to over-segmentation and reflections. Fourth frame contains long track of distant taillights. Better viewed in color. Please, visit *http://www.cvc.uab.es/adas/IHC/ITSC2010/* for complete videos.

REFERENCES

[1] "Traffic safety facts," White Paper, U.S. National Highway Traffic Safety Administration, 2000.

[2] "Use of high-beam headlamps," White Paper, Transportation Research Insitute, Michigan University, 2006.

[3] Y.-L. Chen, C.-T. Lin, C.-J. Fan, C.-M. Hsieh, and B.-F. Wu, "Vision-based nighttime vehicle detection and range estimation for driver assistance," in *SMC'08: Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, oct. 2008, pp. 2988–2993.

[4] R. O'Malley, M. Galvin, and E. Jones, "Vehicle detection at night based on tail-light detection," in *ISVCS'08: Proc. of the 1st Int. Symp. on Vehicular Computing Systems*, 2008.

[5] N. Alt, C. Claus, and W. Stechele, "Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments," in *DATE'08: Proc. of the conf. on Design, Automation and Test i Europe*. New York, NY, USA: ACM, 2008, pp. 176–181.

[6] J. Firl, M. Hoerter, M. Lauer, and C. Stiller, "Vehicle detection, classification and position estimation based on monocular video data during night-time," in *ISAL'09: Proc. of the 8th Int. Symposium on Automotive Lighting*. Utz, München, 2009.

[7] R. DeFauw, S. Lakshmanan, and K. Prasad, "A system for small target detection, tracking, and classification," in *ITSC'99: Proc. of then IEEE Int. Conf. on Intelligent Transportation Systems*, 1999, pp. 639–644.

[8] A. Fossati, P. Schönmann, and P. Fua, "Real-time vehicle tracking for driving assistance," *Machine Vision and Applications*, vol. 28, no. 10, 2010.

[9] Y. Li and S. Pankanti, "Intelligent headlight control using camera sensors," in *UCVP'09: Proc. of the Workshop on Use of Context in Vision Processing*. ACM, 2009, pp. 1–6.

[10] J. Rebut, B. Bradai, J. Moizard, and A. Charpentier, "A monocular vision based advanced lighting automation system for driving assistance," in *ISIE'09: IEEE Int. Symp. on Industrial Electronics*, 2009, pp. 311–316.

[11] P. Alcantarilla *et al.*, "Night time vehicle detection for driving assistance lightbeam controller," in *Intelligent Vehicles Symposium, 2008 IEEE*, 2008, pp. 291–296.

[12] S. Göormer, D. Müller, S. Hold, M. Meuter, and A. Kummert, "Vehicle recognition and ttc estimation at night based on spotlight pairing," in *ITSC'09: Proc. of the IEEE Intelligent Transportation Systems Conf.*, 2009, pp. 1–6.

[13] A. López *et al.*, "Nighttime vehicle detection for intelligent headlight control," in *ACIVS'08: Proc. of 10th Int. Conf. on Advanced Concepts for Intelligent Vision Systems*. Springer-Verlag, 2008, pp. 113–124.

[14] ——, "Temporal coherence analysis for intelligent headlight control," in *Proc. of the IROS'08 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, 2008, pp. 59–64.

[15] Bernard, *Density Estimation for Statistics and Data Analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*, 1st ed. Chapman and Hall/CRC, 1986.

[16] F. Kschischang, S. Member, B. J. Frey, and H. andrea Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, pp. 498–519, 2001.

[17] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs." 2001.

[18] C. Rother, P. Kohli, W. Feng, and J. Jia, "Minimizing sparse higher order energy functions of discrete variables," in *CVPR'09 Proc. of the conf. on Computer Vision and Pattern Recognition*, 2009, pp. 1382–1389.

[19] J. M. M. et al., "libDAI 0.2.4: A free/open source C++ library for Discrete Approximate Inference," http://www.libdai.org/, 2010.

[20] (2010) Tracking videos web page. http://www.cvc.uab.es/adas/IHC/ITSC2010/.